

BBBBBBBBBBBBBB	AAAAAAA	SSSSSSSSSS	RRRRRRRRRR	TTTTTTTTTT	LLL			
BBBBBBBBBBBBBB	AAAAAAA	SSSSSSSSSS	RRRRRRRRRR	TTTTTTTTTT	LLL			
BBBBBBBBBBBBBB	AAAAAAA	SSSSSSSSSS	RRRRRRRRRR	TTTTTTTTTT	LLL			
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBBBBBBBBBBBBB	AAA	AAA	SSSSSSSS	RRRRRRRRRR	TTT	LLL		
BBBBBBBBBBBBBB	AAA	AAA	SSSSSSSS	RRRRRRRRRR	TTT	LLL		
BBBBBBBBBBBBBB	AAA	AAA	SSSSSSSS	RRRRRRRRRR	TTT	LLL		
BBB	BBB	AAAAAA	AAAAAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAAAAA	AAAAAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAAAAA	AAAAAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBB	BBB	AAA	AAA	SSS	RRR	RRR	TTT	LLL
BBBBBBBBBBBBBB	AAA	AAA	SSSSSSSSSS	RRR	RRR	TTT	LLL	
BBBBBBBBBBBBBB	AAA	AAA	SSSSSSSSSS	RRR	RRR	TTT	LLL	
BBBBBBBBBBBBBB	AAA	AAA	SSSSSSSSSS	RRR	RRR	TTT	LLL	

FILEID**BASMATRN

H 14

```

BBBBBBBBBB  AAAA     SSSSSSSS  MM  MM  AAAA     TTTTTTTT  TTTTTTTT  RRRRRRRR  NN  NN
BBBBBBBBBB  AAAA     SSSSSSSS  MM  MM  AAAA     TTTTTTTT  TTTTTTTT  RRRRRRRR  NN  NN
BB  BB  AA  AA  SS  MMMM  MMMM  AA  AA  TT  TT  RR  RR  NN  NN
BB  BB  AA  AA  SS  MMMM  MMMM  AA  AA  TT  TT  RR  RR  NN  NN
BB  BB  AA  AA  SS  MM  MM  MM  AA  AA  AA  TT  TT  RR  RR  NNNN  NN
BB  BB  AA  AA  SS  MM  MM  MM  AA  AA  AA  TT  TT  RR  RR  NNNN  NN
BBBBBBBBBB  AA  AA  SSSSSS  MM  MM  AA  AA  TT  TT  TT  RRRRRRRR  NN  NN
BBBBBBBBBB  AA  AA  SSSSSS  MM  MM  AA  AA  TT  TT  TT  RRRRRRRR  NN  NN
BB  BB  AAAAAAAA  SS  MM  MM  AAAAAAAA  TT  TT  RR  RR  NN  NNNN
BB  BB  AAAAAAAA  SS  MM  MM  AAAAAAAA  TT  TT  RR  RR  NN  NNNN
BB  BB  AA  AA  SS  MM  MM  AA  AA  TT  TT  RR  RR  NN  NN
BB  BB  AA  AA  SS  MM  MM  AA  AA  TT  TT  RR  RR  NN  NN
BBBBBBBBBB  AA  AA  SSSSSSSS  MM  MM  AA  AA  TT  TT  RR  RR  NN  NN
BBBBBBBBBB  AA  AA  SSSSSSSS  MM  MM  AA  AA  TT  TT  RR  RR  NN  NN

```

•
•
•
•
•

(2) 63 DECLARATIONS
(4) 285 BASS\$MAT_TRN - Transpose one matrix into another

0000 1 .TITLE BASSMAT_TRN
0000 2 .IDENT /1-013/
0000 3 ; File: BASMAATTRN.MAR Edit: SBL1013
0000 4
0000 5 *****
0000 6 *
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10 *
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17 *
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21 *
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *
0000 27 ;
0000 28 *
0000 29 **
0000 30 : FACILITY: BASIC code support
0000 31 :
0000 32 : ABSTRACT:
0000 33 :
0000 34 : This module writes the transpose of a matrix into a second matrix.
0000 35 :
0000 36 : ENVIRONMENT: User Mode, AST Reentrant
0000 37 :
0000 38 :--
0000 39 : AUTHOR: R. Will, CREATION DATE: 10-Jul-79
0000 40 :
0000 41 : MODIFIED BY:
0000 42 :++
0000 43 : 1-001 - Original
0000 44 : 1-002 - Fix test for 'same array' for virtual. RW 15-Feb-1980
0000 45 : 1-003 - Add support for byte, g & h floating. PLL 22-Sep-81
0000 46 : 1-004 - More modifications for new data types. PLL 24-Sep-81
0000 47 : 1-005 - Change shared external references to G^ RNH 25-Sep-81
0000 48 : 1-006 - Substitute a macro for the calls to the array fetch and store
0000 49 : routines. This should speed things up. PLL 9-Nov-81
0000 50 : 1-007 - STORE macro must handle g & h floating. PLL 11-Nov-81
0000 51 : 1-008 - Correct a run-time expression in the FETCH and STORE macros.
0000 52 : PLL 20-Jan-82
0000 53 : 1-009 - Don't list macro expansions. PLL 16-Mar-82
0000 54 : 1-010 - Remove FETCH and STORE macros; they are now located in macro
0000 55 : library MATRIXMAC.OLB. Also added support for arrays of
0000 56 : descriptors. LB 19-May-82
0000 57 : 1-011 - Change own storage to stack storage. LB 9-Jul-1982

0000 58 : 1-012 - Allow gfloat results to be stored in a double destination, and
0000 59 : vice versa. PLL 7-Oct-1982
0000 60 : 1-013 - Use G^ for ALL externals. SBL 16-Nov-1982
0000 61 ;--

```

0000 63 .SBTTL DECLARATIONS
0000 64 :
0000 65 : INCLUDE FILES:
0000 66 :
0000 67 :
0000 68 $DSCDEF
0000 69 $SFDEF
0000 70
0000 71 :
0000 72 : EXTERNAL DECLARATIONS:
0000 73 :
0000 74 .DSABL GBL
0000 75
0000 76
0000 77
0000 78 .EXTRN BASSK_ARGDONMAT
0000 79
0000 80 .EXTRN BASSK_DATTYPERR
0000 81
0000 82 .EXTRN BASSK_MATDIMERR
0000 83
0000 84 .EXTRN BASSK_ILOPE
0000 85
0000 86 .EXTRN BASSSTO_FA_B_R8
0000 87 .EXTRN BASSSTO_FA_W_R8
0000 88 .EXTRN BASSSTO_FA_L_R8
0000 89 .EXTRN BASSSTO_FA_F_R8
0000 90 .EXTRN BASSSTO_FA_D_R8
0000 91 .EXTRN BASSSTO_FA_G_R8
0000 92 .EXTRN BASSSTO_FA_H_R8
0000 93 .EXTRN BASSFET_FA_B_R8
0000 94 .EXTRN BASSFET_FA_W_R8
0000 95 .EXTRN BASSFET_FA_L_R8
0000 96 .EXTRN BASSFET_FA_F_R8
0000 97 .EXTRN BASSFET_FA_D_R8
0000 98 .EXTRN BASSFET_FA_G_R8
0000 99 .EXTRN BASSFET_FA_H_R8
0000 100 .EXTRN BASSMAT_REDIM
0000 101
0000 102
0000 103 .EXTRN BASS$SCALE_R1
0000 104 .EXTRN MTH$DINT_R4
0000 105 .EXTRN BASS$STOP
0000 106 .EXTRN BASSFETCH_BFA
0000 107 .EXTRN BASSSTORE_BFA
0000 108
0000 109
0000 110 :
0000 111 : MACROS:
0000 112 :
0000 113 :
0000 114 : SBASSMAT_TRN transpose loop algorithm, see next page
0000 115 : FETCH fetch an element from an array (found in macro
0000 116 : library MATRIXMAC.OLB)
0000 117 : STORE store an element into an array (found in macro
0000 118 : library MATRIXMAC.OLB)
0000 119 :

```

0000 120 :
0000 121 : EQUATED SYMBOLS:
0000 122 :
0000 123 :
00000004 0000 124 first_arg = 4 ; arg offset for str copy
00000008 0000 125 second_arg = 8 ; arg offset for str copy
0000000C 0000 126 index1 = 12 ; stack offset for str copy
00000010 0000 127 index2 = 16 ; stack offset for str copy
00000014 0000 128 temp_desc = 20 ; stack offset for str copy
00000000 0000 129 lower_bnd2 = 0 ; stack offset for temp
00000004 0000 130 lower_bnd1 = 4 ; stack offset for temp
00000008 0000 131 upper_bnd1 = 8 ; stack offset for temp
0000000C 0000 132 value_desc = 12 ; output descriptor
0000000C 0000 133 str_len = 12 ; length field in desc
0000000E 0000 134 dtype = 14 ; data type field in desc
0000000F 0000 135 class = 15 ; class field within desc
00000010 0000 136 pointer = 16 ; pointer to data
00000014 0000 137 data = 20 ; data
00000018 0000 138 dsc\$l_l1_1 = 24 ; desc offset if 1 sub
0000001C 0000 139 dsc\$l_u1_1 = 28 ; desc offset if 1 sub
0000001C 0000 140 dsc\$l_l1_2 = 28 ; desc offset if 2 sub
00000020 0000 141 dsc\$l_u1_2 = 32 ; desc offset if 2 sub
00000024 0000 142 dsc\$l_l2_2 = 36 ; desc offset if 2 sub
00000028 0000 143 dsc\$l_u2_2 = 40 ; desc offset if 2 sub
0000 144 :
0000 145 :
0000 146 : OWN STORAGE:
0000 147 :
0000 148 :
0000 149 :
0000 150 :
0000 151 : PSECT DECLARATIONS:
0000 152 :
00000000 153 .PSECT _BASS\$CODE PIC, USR, CON, REL, LCL, SHR, -
0000 154 EXE, RD, NOWRT, LONG
0000 155

```

0000 157 ;+
0000 158 ; This macro contains the looping mechanism for accessing all elements of
0000 159 ; an array. It also contains all the logic for all the combinations of data
0000 160 ; types and scaling. A macro is used to make it easy to maintain the parallel
0000 161 ; code for all the different data types.
0000 162 ;-
0000 163
0000 164 .MACRO $BASS$MAT_TRN src_dtype, dest_dtype ; transpose algorithm
0000 165
0000 166 ;+
0000 167 ; Loop through all the rows. Row and column upper and lower bounds have been
0000 168 ; initialized on the stack.
0000 169 ;-
0000 170
0000 171 LOOP_1ST_SUB'src_dtype'dest_dtype':
0000 172     MOVL [lower_bnd2(SP), R11] ; R11 has 2nd lower bound
0000 173
0000 174 ;+
0000 175 ; Loop through all the elements (columns) of the current row. Column lower
0000 176 ; bound is initialized in R11. Column upper bound is on the stack.
0000 177 ; Distinguish array by data type so that the correct fetch routine can
0000 178 ; retrieve the data, the correct conversion can be done and the correct
0000 179 ; store routine can be called.
0000 180 ;-
0000 181
0000 182 LOOP_2ND_SUB'src_dtype'dest_dtype':
0000 183
0000 184 ;+
0000 185 ; Get the data from source array
0000 186 ;-
0000 187
0000 188     MOVL R10, R0 ; pointer to array dest
0000 189     MOVL lower_bnd1(SP), R1 ; current row
0000 190     MOVL R11, R2 ; current col
0000 191     FETCH 'src_dtype' ; fetch data from src array
0000 192
0000 193 ;+
0000 194 ; If the data types of the source and destination arrays is different,
0000 195 ; convert the data to the destination type. If scaling is needed (ie if
0000 196 ; at least one but not both of the arrays is double) scale the data.
0000 197 ;-
0000 198
0000 199     .IF DIF src_dtype, dest_dtype ; src and dest arrays are not
0000 200 ; save data type
0000 201     .IF IDN src_dtype, G ; source is gfloat
0000 202     .IF IDN dest_dtype, D ; don't try to CVTGD
0000 203     CVTGH R0, R0 ; promote source to hfloat
0000 204     .IFF
0000 205     CVT'src_dtype'dest_dtype' R0, R0 ; dest is not dbl
0000 206     .ENDC ; OK to cvt to dest type
0000 207     .IFF
0000 208     .IF IDN src_dtype, D ; source is double
0000 209     MOVD R0, -(SP) ; save the data
0000 210     MOVL SF$L SAVE FP(FP), R0 ; pass FP to get scale
0000 211     JSB G$BASS$SCALE_R1 ; get scale in R0 & R1
0000 212 ; call a BLISS routine because
0000 213 ; the frame offsets are only

```

```

0000 214
0000 215 DIVD3 R0, (SP)+, R0 ; defined for BLISS
0000 216 .IF IDN dest_dtype, G ; scale
0000 217 CVTDH R0, R0 ; can't CVTDG
0000 218 .IFF dest is not gfloat
0000 219 CVT'src_dtype'dest_dtype' R0, R0 ; so promote to hfloat
0000 220 .ENDC ; dest is not gfloat
0000 221 .IFF
0000 222 CVT'src_dtype'dest_dtype' R0, R0 ; cvt src to dest type
0000 223 .IF IDN dest_dtype, D ; convert data from R0 into R0
0000 224 MOVD R0, -(SP) ; dest is double
0000 225 MOVL SF$L SAVE_FP(FP), R0 ; save the data
0000 226 JSB G^BASSSCALE_R1 ; pass FP to get scale
0000 227
0000 228
0000 229
0000 230 MULD2 (SP)+, R0 ; get scale in R0 & R1
0000 231 JSB G^MTH$DINT_R4 ; call a BLISS routine because
0000 232 .ENDC ; the frame offsets are only
0000 233 .ENDC ; defined for BLISS
0000 234 .ENDC ; scale
0000 235 .ENDC ; integerize
0000 236
0000 237 :+
0000 238 : Now store the data in the destination array.
0000 239 : Hfloat passed by value takes 4 words, gfloat and double take 2 words, and
0000 240 : all other supported daty types take 1 longword.
0000 241 :-+
0000 242
0000 243 .IF IDN dest_dtype, H ; dtype is hfloat
0000 244 MOVL dest_matrix(AP), R4 ; pointer to array desc
0000 245 MOVL lower_bnd1(SP), R6 ; current row, put in col
0000 246 MOVL R11, R5 ; current col, put in row
0000 247 .IFF
0000 248 .IF IDN dest_dtype, G ; dtype is gfloat
0000 249 MOVL dest_matrix(AP), R2 ; pointer to array desc
0000 250 MOVL lower_bnd1(SP), R4 ; current row, put in col
0000 251 MOVL R11, R3 ; current col, put in row
0000 252 .IFF
0000 253 .IF IDN dest_dtype, D ; see if dtype is double
0000 254 MOVL dest_matrix(AP), R2 ; pointer to array desc
0000 255 MOVL lower_bnd1(SP), R4 ; current row, put in col
0000 256 MOVL R11, R3 ; current column, put in row
0000 257 .IFF
0000 258 MOVL dest_matrix(AP), R1 ; all other data types here
0000 259 MOVL lower_bnd1(SP), R3 ; pointer to array desc
0000 260 MOVL R11, R2 ; current row, put in col
0000 261 .ENDC ; current col, put in row
0000 262 .ENDC
0000 263 .ENDC
0000 264 MOV'dest_dtype' R0, DATA(SP) ; code now same for all dtypes
0000 265 STORE 'dest_dtype' ; store value in DATA
0000 266 INCL R11 ; store in array
0000 267 CMPL R11, R9 ; get next column
0000 268 BGTR 3S ; see if last column done
0000 269 BRW LOOP_2ND_SUB'src_dtype'dest_dtype' ; no, continue inner loop
0000 270

```

```
0000 271 :+
0000 272 : Have completed entire row. See if it was the last row. If not,
0000 273 : continue with next row.
0000 274 ;-
0000 275
0000 276 3$:    INCL    lower_bnd1(SP)           ; get next row
0000 277     CMPL    lower_bnd1(SP), upper_bnd1(SP) ; see if last row done
0000 278     BGTR    5$                            ; loop
0000 279     BRW     LOOP_1ST_SUB'src_dtype'dest_dtype' ; no, continue outer loop
0000 280
0000 281 5$:    RET
0000 282
0000 283     .ENDM
```

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:35:09 VAX/VMS Macro V04-
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMAT

```

0000 285 .SBTTL BASS$MAT_TRN - Transpose one matrix into another
0000 286 ++
0000 287 : FUNCTIONAL DESCRIPTION:
0000 288 :
0000 289 : Transpose one matrix into another. If the src matrix has 2 dimensions,
0000 290 : redimension the output matrix to have the number of rows that the src
0000 291 : has columns and the number of columns that the src has rows. (thereby
0000 292 : ensuring that the dest matrix also has 2 dimensions). Initialize all
0000 293 : the necessary looping information on the stack. Conversions will have
0000 294 : to be done from the source data type to the destination data type, so
0000 295 : divide the looping portion according to the data types.
0000 296 :
0000 297 : CALLING SEQUENCE:
0000 298 :
0000 299 : CALL BASMAT_TRN (src_matrix.rx.da, dest_matrix.wx.da)
0000 300 :
0000 301 : INPUT PARAMETERS:
0000 302 :
0000 303 : src_matrix = 4
0000 304 :
0000 305 : IMPLICIT INPUTS:
0000 306 :
0000 307 : Scale from the callers frame to scale double precision.
0000 308 :
0000 309 : OUTPUT PARAMETERS:
0000 310 :
0000 311 : dest_matrix = 8
0000 312 :
0000 313 : IMPLICIT OUTPUTS:
0000 314 :
0000 315 : NONE
0000 316 :
0000 317 : FUNCTION VALUE:
0000 318 : COMPLETION CODES:
0000 319 :
0000 320 : NONE
0000 321 :
0000 322 : SIDE EFFECTS:
0000 323 :
0000 324 : This routine calls the redimensioning routine and the array element
0000 325 : fetch and store routines and therefore may signal any of their errors.
0000 326 : It may also signal any of the errors listed in the externals section.
0000 327 : It may also cause the destination array to have different dimensions.
0000 328 :
0000 329 :--
0000 330 :
0000 331 .ENTRY BASS$MAT_TRN, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,IV>
0002 332 :
0002 333 :+
0002 334 : REGISTER USAGE
0002 335 : R0 - R8 destroyed by fetch and store routines
0002 336 : R9 upper bound for 2nd subscript
0002 337 : R10 pointer to array descriptor
0002 338 : R11 current value of 2nd subscript
0002 339 :-
0002 340 :
0002 341 :+

```

BASSMAT_TRN - Transpose one matrix into

0002 342 : Put routine arguments into registers for ease of use.
 0002 343 : If block 3 of array descriptor (bounds) is not present then error.
 0002 344 :-
 0002 345
 7E 7C 0002 346 CLRQ -(SP)
 7E 7C 0004 347 CLRQ -(SP)
 7E 7C 0006 348 CLRQ -(SP)
 5A 04 AC 0008 349 MOVL src_matrix(AP), R10 ; ptr to array descr in R10
 13 0A AA 07 E1 000C 350 BBC #DSC\$V_FL_BOUNDS, DSC\$B_AFLAGS(R10), ERR_ARGDONMAT
 0011 351 ; exit if block 3 not
 0011 352 ; present in descriptor
 0011 353
 0011 354 ;+
 0011 355 : Set up limits for looping through all elements
 0011 356 :-
 0011 357
 02 08 AA 91 0011 358 CMPB DSC\$B_DIMCT(R10), #2 ; determine # of subscripts
 27 13 0015 359 BEQLU INIT_SUBS ; 2 subs, initialize loop
 0017 360 ; not 2 subs, fall into error
 0017 361
 0017 362 ERR_MATDIMERR:
 00000000'8F 0017 363 PUSHL #BAS\$K_MATDIMERR ; signal error, not 2 dims
 00000000'GF 01 FB 001D 364 CALLS #1, G^BASS\$STOP ; for src matrix
 0024 365
 0024 366 ERR_ARGDONMAT:
 00000000'8F 0024 367 PUSHL #BAS\$K_ARGDONMAT ; signal error,
 00000000'GF 01 FB 002A 368 CALLS #1, G^BASS\$STOP ; block 2 or 3 absent
 0031 369
 0031 370 ERR_ILOPE:
 00000000'8F 0031 371 PUSHL #BAS\$K_ILOPE ; signal error, DSC\$A_POINTER
 00000000'GF 01 FB 0037 372 CALLS #1, G^BASS\$STOP ; same for src & dest matrices
 003E 373
 003E 374 ;+
 003E 375 : There are 2 subscripts. Make sure the source and dest matrices are not the
 003E 376 : same. Check and redimension the destination array if
 003E 377 : necessary. Put the upper bound for both subscripts on the
 003E 378 : stack and make sure that the lower bound for both subscripts will start
 003E 379 : at 1 (do not alter row or col 0)
 003E 380 :-
 003E 381
 003E 382 INIT_SUBS:
 5B 08 AC D0 003E 383 MOVL dest_matrix(AP), R11 ; get pointer to dest matrix
 04 03 AA 91 0042 384 CMPB DSC\$B_CLASS(R10), #DSC\$K_CLASS_A ; is src virtual
 04 0F 12 0046 385 BNEQU VIRTUAL ; src is virtual go process
 04 03 AB 91 0048 386 CMPB DSC\$B_CLASS(R11), #DSC\$K_CLASS_A ; is dest virtual?
 04 1D 12 004C 387 BNEQU INIT_SUBS_2 ; yes, cant be same array
 04 AB 04 AA D1 004E 388 CMPL DSC\$A_POINTER(R10), DSC\$A_POINTER(R11) ; are matrices the same?
 DC 13 0053 389 BEQLU ERR_ILOPE ; yes, error
 14 11 0055 390 BRB INIT_SUBS_2
 003E 391 VIRTUAL:
 04 03 AB 91 0057 392 CMPB DSC\$B_CLASS(R11), #DSC\$K_CLASS_A ; is dest virtual
 04 0E 13 005B 393 BEQLU INIT_SUBS_2 ; no, cant be same array
 FC AB FC AA D1 005D 394 CMPL DSC\$C_LOGUNIT(R10), DSC\$L_LOGUNIT(R11) ; are matrices same
 07 12 0062 395 BNEQ INIT_SUBS_2 ; no
 F8 AB F8 AA D1 0064 396 CMPL DSC\$C_BYTEOFF(R10), DSC\$L_BYTEOFF(R11) ; are the matrices
 0069 397 ; the same, (test for same by
 0069 398 ; same dsc\$C_byteoff, dsc\$L_logunit

BASSMAT_TRN - Transpose one matrix into

00000000'GF 03 FB 0073 404 CALLS #3, G^BASSMAT REDIM
 20 AA DD 006B 401 PUSHL dsc\$l_u1_2(R10)
 28 AA DD 006E 402 PUSHL dsc\$l_u2_2(R10)
 5B DD 0071 403 PUSHL R11
 03 FB 0073 404 CALLS #3, G^BASSMAT REDIM
 20 AA DD 007A 405 PUSHL dsc\$l_u1_2(R10)
 1C AA DD 007D 406 PUSHL dsc\$l_l1_2(R10)
 03 14 0080 407 BGTR 1\$
 59 6E 01 DO 0082 408 MOVL #1, (SP)
 28 AA DO 0085 409 1\$: MOVL dsc\$l_u2_2(R10), R9
 24 AA DD 0089 410 PUSHL dsc\$l_l2_2(R10)
 03 14 008C 411 BGTR SEPARATE_DTYPES
 6E U1 DO 008E 412 MOVL #1, (SP)
 0091 413
 0091 414 :+
 0091 415 : Algorithm now differs according to data types
 0091 416 :-
 0091 417
 0091 418 SEPARATE_DTYPES:
 05 06 55 5A DO 0091 419 MOVL R10, R5
 8F 0094 420 5\$: CASEB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_B, #<DSC\$K_DTYPE_D - DSC\$K_DTYPE_B>
 0037' 0099 421 2\$: .WORD BYTE-2\$
 0DB2' 009B 422 .WORD WORD-2\$
 1B2D' 009D 423 .WORD LONG-2\$
 002A' 009F 424 .WORD ERR_DATTYPERR-2\$
 28AB' 00A1 425 .WORD FLOAT-2\$
 3623' 00A3 426 .WORD DOUBLE-2\$
 00A5 427
 00A5 428 :+
 00A5 429 : G and H floating fall outside the range of the CASEB, so check for them
 00A5 430 : separately.
 00A5 431 :-
 00A5 432
 1B 02 A5 91 00A5 433 CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_G
 03 12 00A9 434 BNEQ 3\$
 43D9 31 00AB 435 BRW GFLOAT
 1C 02 A5 91 00AE 436
 03 12 00B2 437 3\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_H
 5163 31 00B4 438 BNEQ 4\$
 00B7 439 BRW HFLOAT
 18 02 A5 91 00B7 440
 06 12 00BB 441 4\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_DSC
 55 04 A5 DO 00BD 442 BNEQ ERR_DATTYPERR
 D1 11 00C1 443 MOVL 4(R5), R5
 00C3 444 BRB 5\$
 00C3 445
 00000000'BF 01 FB 00C9 446 ERR_DATTYPERR:
 00000000'GF 01 DD 00C3 447 PUSHL #BAS\$K_DATTYPERR
 00000000'GF 01 FB 00C9 448 CALLS #1, G^BASS\$STOP

: yes, error
 : 2nd upr bnd, make 1st in dest
 : 1st upr bnd, make 2nd in dest
 : dest array pointer
 : redimension destination
 : 1st upper bound
 : 1st lower bound
 : not row 0 or neg, do cols
 : start with row 1
 : 2nd upper bound
 : 2nd lower bound
 : not col 0 or neg, go loop
 : start with col 1

: save original pointer
 : code for byte dtype
 : code for word dtype
 : code for long dtype
 : quad not supported
 : code for float dtype
 : code for double dtype

: R5 <-- addr of descriptor
 : CASE again on dtype in desc

: Signal error, unsupported
 : dtype in array desc

0000 451 ;+
 0000 452 ; Source array is a byte array. Now differentiate on the destination type.
 0000 453 ;-
 0000 454
 05 06 55 5B D0 0000 455 BYTE: MOVL R11, R5 ; move pointer into R5
 8F 00D3 456 5\$: CASEB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_B, #<DSC\$K_DTYPE_D - DSC\$K_DTYPE_B>
 002D 00D8 457 1\$: WORD BYTE_TO_BYTE-1\$; code for byte dtype
 020B 00DA 458 WORD BYTE_TO_WORD-1\$; code for word dtype
 03EC 00DC 459 WORD BYTE_TO_LONG-1\$; code for long dtype
 FFEB 00DE 460 WORD ERR_DATTYPERR-1\$; quad not supported
 05CD 00E0 461 WORD BYTE_TO_FLOAT-1\$; code for float dtype
 07AE 00E2 462 WORD BYTE_TO_DOUBLE-1\$; code for double dtype
 00E4 463
 1B 02 A5 91 00E4 464 CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_G
 03 12 00E8 465 BNEQ 2\$
 0990 31 00EA 466 BRW BYTE_TO_GFLOAT
 00ED 467
 1C 02 A5 91 00ED 468 2\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_H
 03 12 00F1 469 BNEQ 3\$
 0B6E 31 00F3 470 BRW BYTE_TO_HFLOAT
 00F6 471
 18 02 A5 91 00F6 472 3\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_DSC
 06 12 00FA 473 BNEQ 4\$
 55 04 A5 D0 00FC 474 MOVL 4(R5), R5 ; R5 <- addr of descriptor
 D1 11 0100 475 BRB 5\$; CASE again on dtype in desc
 0102 476
 FFBF 31 0102 477 4\$: BRW ERR_DATTYPERR ; unsupported dtype
 0105 478
 0105 479 ;+
 0105 480 ; Now type of source and destination arrays are known. Use the macro to
 0105 481 ; generate the code for each case
 0105 482 ;-
 0105 483

BASS\$MAT_TRN
1-013

H 15

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 12
(5)

0105 485 BYTE_TO_BYT: \$BASS\$MAT_TRN B, B
02E3 486

BASSMAT_TRN
1-013

I 15

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMATTRN.MAR;1 Page 13
(5)

02E3 488 BYTE_TO_WORD: \$BASSMAT_TRN B, W
04C4 489

BASSMAT_TRN
1-013

J 15

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 14
(5)

04C4 491 BYTE_TO_LONG: \$BASSMAT_TRN B, L
06A5 492

BASSMAT_TRN
1-013

K 15

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATRN.MAR;1 Page 15
(5)

06A5 494 BYTE_TO_FLOAT: \$BASSMAT_TRN B, F
0886 495

BASSMAT_TRN
1-013

L 15

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 16
(5)

0886 497 BYTE_TO_DOUBLE: \$BASSMAT_TRN B, D
0A7D 498

BASSMAT_TRN
1-013

M 15

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 17
(5)

0A7D 500 BYTE_TO_GFLOAT: \$BASSMAT_TRN B, G
0C64 501

BASSMAT_TRN
1-013

N 15

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 18
(5)

0C64 503 BYTE_TO_HFLOAT: \$BASSMAT_TRN B, H
0E4B 504

0E4B 506 ;+
 0E4B 507 ; Source array is a word array. Now differentiate on the destination type.
 0E4B 508 ;-
 0E4B 509

05 06 55 5B D0 0E4B 510 WORD: MOVL R11, R5 ; move original pointer in R5
 8F 0E4E 511 5\$: CASEB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_B, #<DSC\$K_DTYPE_D - DSC\$K_DTYPE_B>
 002D 0E53 512 1\$: .WORD WORD_TO_BYTE-1\$; code for byte dtype
 020E 0E55 513 .WORD WORD_TO_WORD-1\$; code for word dtype
 03EC 0E57 514 .WORD WORD_TO_LONG-1\$; code for long dtype
 F270 0E59 515 .WORD ERR_DATTYPERR-1\$; quad not supported
 05CD 0E5B 516 .WORD WORD_TO_FLOAT-1\$; code for float dtype
 07AE 0E5D 517 .WORD WORD_TO_DOUBLE-1\$; code for double dtype
 0E5F 518

1B 02 A5 91 0E5F 519 CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_G
 03 12 0E63 520 BNEQ 2\$
 0990 31 0E65 521 BRW WORD_TO_GFLOAT

1C 02 A5 91 0E68 522
 03 12 0E6C 523 2\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_H
 0B6E 31 0E6E 524 BNEQ 3\$
 0E71 525 BRW WORD_TO_HFLOAT

18 02 A5 91 0E71 526
 06 12 0E75 527 3\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_DSC
 55 04 A5 D0 0E77 528 BNEQ 4\$
 D1 11 0E7B 529 MOVL 4(R5), R5 ; R5 <-- addr of descriptor
 0E7D 530 BRB 5\$; CASE again on dtype in desc

F243 31 0E7D 531
 0E80 532 4\$: BRW ERR_DATTYPERR ; unsupported dtype
 0E80 533
 0E80 534 ;+
 0E80 535 ; Now type of source and destination arrays are known. Use the macro to
 0E80 536 ; generate the code for each case
 0E80 537 ;-
 0E80 538

BASSMAT_TRN
1-013

C 16

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 20
(5)

OE80 540 WORD_TO_BYTE: \$BASSMAT_TRN W, B
1061 541

BASSMAT_TRN
1-013

D 16

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 21
(5)

1061 543 WORD_TO_WORD: \$BASSMAT_TRN W, W
123F 544

BASS\$MAT_TRN
1-013

E 16

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 22
(5)

123F 546 WORD_TO_LONG: \$BASS\$MAT_TRN W, L
1420 547

BASSMAT_TRN
1-013

F 16

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMATTRN.MAR;1 Page 23 (5)

1420 549 WORD_TO_FLOAT: \$BASSMAT_TRN W, F
1601 550

BASS\$MAT_TRN
1-013

G 16

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 24 (5)

1601 552 WORD_TO_DOUBLE: \$BASS\$MAT_TRN W, D
17F8 553

BASSMAT_TRN
1-013

H 16

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMATTRN.MAR;1 Page 25
(5)

17F8 555 WORD_TO_GFLOAT: \$BASSMAT_TRN W, G
19DF 556

BASSMAT_TRN
1-013

I 16

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 26
(5)

19DF 558 WORD_TO_HFLOAT: \$BASSMAT_TRN W, H
1BC6 559

1BC6 561 ;+
 1BC6 562 ; Source array is a longword array. Now differentiate on the destination type
 1BC6 563 ;-
 1BC6 564
 05 06 55 5B D0 1BC6 565 LONG: MOVL R11, R5 ; recover original pointer
 8F 1BC9 566 5\$: CASEB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_B, #<DSC\$K_DTYPE_D - DSC\$K_DTYPE_B>
 002D' 1BCE 567 1\$: .WORD LONG_TO_BYTE-1\$; code for byte dtype
 020E' 1BD0 568 .WORD LONG_TO_WORD-1\$; code for word dtype
 03EF' 1BD2 569 .WORD LONG_TO_LONG-1\$; code for long dtype
 E4F5' 1BD4 570 .WORD ERR_DATTYPERR-1\$; quad not supported
 05CD' 1BD6 571 .WORD LONG_TO_FLOAT-1\$; code for float dtype
 07AE' 1BD8 572 .WORD LONG_TO_DOUBLE-1\$; code for double dtype
 1BDA 573
 1B 02 A5 91 1BDA 574 CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_G
 03 12 1BDE 575 BNEQ 2\$
 0990 31 1BEO 576 BRW LONG_TO_GFLOAT
 1C 02 A5 91 1BE3 577
 03 12 1BE7 578 2\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_H
 0B6E 31 1BE9 579 BNEQ 3\$
 1BEC 580 BRW LONG_TO_HFLOAT
 18 02 A5 91 1BEC 581
 06 12 1BF0 582 3\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_DSC
 55 04 A5 D0 1BF2 583 BNEQ 4\$
 D1 11 1BF6 584 MOVL 4(R5), R5 ; R5 <-- addr of descriptor
 1BF8 585 BRB 5\$; CASE again on dtype in desc
 E4C8 31 1BF8 586
 1BFB 587 4\$: BRW ERR_DATTYPERR ; unsupported dtype
 1BFB 588
 1BFB 589 ;+
 1BFB 590 ; Now type of source and destination arrays are known. Use the macro to
 1BFB 591 ; generate the code for each case
 1BFB 592 ;-

BASSMAT_TRN
1-013

K 16

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATRN.MAR;1 Page 28
1BFB 594 LONG_TO_BYTE: \$BASSMAT_TRN L, B
1DDC 595

BASSMAT_TRN
1-013

L 16

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATRN.MAR;1 Page 29
(5)

1DDC 597 LONG_TO_WORD: \$BASSMAT_TRN L, W
1FBD 598

BASS\$MAT_TRN
1-013

M 16

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 30
(5)

1FB0D 600 LONG_TO_LONG: \$BASS\$MAT_TRN L, L
219B 601

BASS\$MAT_TRN
1-013

B 1

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 31 (5)

219B 603 LONG_TO_FLOAT: \$BASS\$MAT_TRN L, F
237C 604

BASS\$MAT_TRN
1-013

C 1

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 32
(5)

237C 606 LONG_TO_DOUBLE: \$BASS\$MAT_TRN L, D
2573 607

BASS\$MAT_TRN
1-013

D 1

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 33 (5)

2573 609 LONG_TO_GFLOAT: \$BASS\$MAT_TRN L, G
275A 610

BASS\$MAT_TRN
1-013

E 1

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 34
(5)

275A 612 LONG_TO_HFLOAT: \$BASS\$MAT_TRN L, H
2941 613

```

2941 615 :+
2941 616 : Source array is a floating array. Now differentiate on the destination type
2941 617 :-
2941 618
05 06 55 5B 00 2941 619 FLOAT: MOVL R11, R5 ; recover original pointer
06 02 A5 8F 2944 620 5$: CASEB DSC$B_DTYPE(R5), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
002D 2949 621 1$: .WORD FLOAT_TO_BYTE-1$ ; code for byte dtype
020E 294B 622 .WORD FLOAT_TO_WORD-1$ ; code for word dtype
03EF 294D 623 .WORD FLOAT_TO_LONG-1$ ; code for long dtype
D77A 294F 624 .WORD ERR_DATTYPERR-1$ ; quad not supported
05D0 2951 625 .WORD FLOAT_TO_FLOAT-1$ ; code for float dtype
07AE 2953 626 .WORD FLOAT_TO_DOUBL-1$ ; code for double dtype
2955 627
1B 02 A5 91 2955 628 CMPB DSC$B_DTYPE(R5), #DSC$K_DTYPE_G
03 12 2959 629 BNEQ 2$ ; code for generic dtype
0990 31 295B 630 BRW FLOAT_TO_GFLOA
1C 02 A5 91 295E 631
03 12 2962 632 2$: CMPB DSC$B_DTYPE(R5), #DSC$K_DTYPE_H
0B6E 31 2964 633 BNEQ 3$ ; code for halfword dtype
2967 634 BRW FLOAT_TO_HFLOA
18 02 A5 91 2967 635
06 12 296B 636 3$: CMPB DSC$B_DTYPE(R5), #DSC$K_DTYPE_DSC
296D 637 BNEQ 4$ ; code for double-precision dtype
55 04 A5 D0 296D 638 MOVL 4(R5), R5 ; R5 <-- addr of descriptor
D1 11 2971 639 BRB 5$ ; CASE again on dtype in desc
2973 640
D74D 31 2973 641 4$: BRW ERR_DATTYPERR ; unsupported dtype
2976 642
2976 643 :+
2976 644 : Now type of source and destination arrays are known. Use the macro to
2976 645 : generate the code for each case
2976 646 :-

```

BASS\$MAT_TRN
1-013

G 1

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 36
(5)

2976 648 FLOAT_TO_BYTE: \$BASS\$MAT_TRN F, B
2857 649

BASS\$MAT_TRN
1-013

H 1
BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 37
(5)
2B57 651 FLOAT_TO_WORD: \$BASS\$MAT_TRN F, W
2D38 652

BASS\$MAT_TRN
1-013

I 1

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 38
(5)

2D38 654 FLOAT_TO_LONG: \$BASS\$MAT_TRN F, L
2F19 655

BASS\$MAT_TRN
1-013

J 1

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 39
(5)

2F19 657 FLOAT_TO_FLOAT: \$BASS\$MAT_TRN F, F
30F7 658

BASS\$MAT_TRN
1-013

K 1

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 40
(5)

30F7 660 FLOAT_TO_DOUBL: \$BASS\$MAT_TRN F, D
32EE 661

BASS\$MAT_TRN
1-013

L 1

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 41
(5)

32EE 663 FLOAT_TO_GFLOA: \$BASS\$MAT_TRN F, G
34D5 664

BASSMAT_TRN
1-013

M 1

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMATTRN.MAR;1 Page 42
(5)

34D5 666 FLOAT_TO_HFLOA: \$BASSMAT_TRN F, H
36BC 667

36BC 669 :+
 36BC 670 : Source array is a double array. Now differentiate on the destination type.
 36BC 671 :-
 36BC 672
 05 06 55 5B D0 36BC 673 DOUBLE: MOVL R11, R5 ; recover original pointer
 8F 36BF 674 5\$: CASEB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_B, #<DSC\$K_DTYPE_D - DSC\$K_DTYPE_B>
 002D' 36C4 675 1\$: .WORD DOUBLE_TO_BYTE-1\$; code for byte dtype
 021F' 36C6 676 .WORD DOUBLE_TO_WORD-1\$; code for word dtype
 0411' 36C8 677 .WORD DOUBLE_TO_LONG-1\$; code for long dtype
 C9FF' 36CA 678 .WORD ERR_DATTYPERR-1\$; quad not supported
 0603' 36CC 679 .WORD DOUBLE_TO_FLOAT-1\$; code for float dtype
 07F5' 36CE 680 .WORD DOUBLE_TO_DOUBL-1\$; code for double dtype
 36D0 681
 1B 02 A5 91 36D0 682 CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_G
 03 12 36D4 683 BNEQ 2\$
 09BE 31 36D6 684 BRW DOUBLE_TO_GFLOA
 36D9 685
 1C 02 A5 91 36D9 686 2\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_H
 03 12 36DD 687 BNEQ 3\$
 0BAD 31 36DF 688 BRW DOUBLE_TO_HFLOA
 36E2 689
 18 02 A5 91 36E2 690 3\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_DSC
 06 12 36E6 691 BNEQ 4\$
 55 04 A5 D0 36E8 692 MOVL 4(R5), R5 ; R5 <-- addr of descriptor
 D1 11 36EC 693 BRB 5\$; CASE again on dtype in desc
 36EE 694
 C9D2 31 36EE 695 4\$: BRW ERR_DATTYPERR ; unsupported dtype
 36F1 696
 36F1 697 :+
 36F1 698 : Now type of source and destination arrays are known. Use the macro to
 36F1 699 : generate the code for each case
 36F1 700 :-

BASS\$MAT_TRN
1-013

B 2

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATRN.MAR;1 Page 44
(5)

36F1 702 DOUBLE_TO_BYTE: \$BASS\$MAT_TRN D, B
38E3 703

B
1

BASS\$MAT_TRN
1-013

C 2

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 45
(5)

38E3 705 DOUBLE_TO_WORD: \$BASS\$MAT_TRN D, W
3AD5 706

B
1

BASSMAT_TRN
1-013

D 2

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMATTRN.MAR;1 Page 46 (5)

3AD5 708 DOUBLE_TO_LONG: \$BASSMAT_TRN D, L
3CC7 709

BASSMAT_TRN
1-013

E 2

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 47 (5)

3CC7 711 DOUBLE_TO_FLOAT: \$BASSMAT_TRN D, F
3EB9 712

BASS\$MAT_TRN
1-013

F 2

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 48
(5)

3EB9 714 DOUBLE_TO_DOUBL: \$BASS\$MAT_TRN D, D
4097 715

BASSMAT_TRN
1-013

G 2

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMATTRN.MAR;1 Page 49 (5)

4097 717 DOUBLE_TO_GFLOA: \$BASSMAT_TRN D, G
428F 718

BASSMAT_TRN
1-013

H 2

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMATTRN.MAR;1 Page 50 (5)

428F 720 DOUBLE_TO_HFLOA: \$BASSMAT_TRN D, H
4487 721

4487 723 :+
 4487 724 : Source array is a gfloat array. Now differentiate on the destination type.
 4487 725 :-
 4487 726
 05 06 55 5B D0 4487 727 GFLOAT: MOVL R11, R5 ; recover original pointer
 8F 448A 728 5\$: CASEB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_B, #<DSC\$K_DTYPE_D - DSC\$K_DTYPE_B>
 002D' 448F 729 1\$: .WORD GFLOAT_TO_BYTE-1\$; code for byte dtype
 0215' 4491 730 .WORD GFLOAT_TO_WORD-1\$; code for word dtype
 03FD' 4493 731 .WORD GFLOAT_TO_LONG-1\$; code for long dtype
 BC34' 4495 732 .WORD ERR_DATTYPERR-1\$; quad not supported
 05E5' 4497 733 .WORD GFLOAT_TO_FLOAT-1\$; code for float dtype
 07CD' 4499 734 .WORD GFLOAT_TO_DOUBL-1\$; code for double dtype
 449B 735
 1B 02 A5 91 449B 736 CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_G
 03 12 449F 737 BNEQ 2\$
 09A0 31 44A1 738 BRW GFLOAT_TO_GFLOA
 44A4 739
 1C 02 A5 91 44A4 740 2\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_H
 03 12 44AB 741 BNEQ 3\$
 0B80 31 44AA 742 BRW GFLOAT_TO_HFLOA
 44AD 743
 18 02 A5 91 44AD 744 3\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_DSC
 06 12 44B1 745 BNEQ 4\$
 55 04 A5 D0 44B3 746 MOVL 4(R5), R5 ; R5 <-- addr of descriptor
 D1 11 44B7 747 BRB 5\$; CASE again on dtype in desc
 44B9 748
 BC07 31 44B9 749 4\$: BRW ERR_DATTYPERR ; unsupported dtype
 44BC 750
 44BC 751 :+
 44BC 752 : Now type of source and destination arrays are known. Use the macro to
 44BC 753 : generate the code for each case
 44BC 754 :-
 44BC 755

BASSMAT_TRN
1-013

J 2

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMATTRN.MAR;1 Page 52 (5)

44BC 757 GFLOAT_TO_BYTE: \$BASSMAT_TRN G, B
46A4 758

BASS\$MAT_TRN
1-013

K 2

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 53
(5)

46A4 760 GFLOAT_TO_WORD: \$BASS\$MAT_TRN G, W
488C 761

BAS\$MAT_TRN
1-013

L 2

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMATTRN.MAR;1 Page 54 (5)

488C 763 GFLOAT_TO_LONG: \$BASSMAT_TRN G, L
4A74 764

BAS\$MAT_TRN
1-013

M 2

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMATTRN.MAR;1 Page 55 (5)

4A74 766 GFLOAT_TO_FLOAT: \$BASSMAT_TRN G, F
4C5C 767

BASS\$MAT_TRN
1-013

N 2

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 56 (5)

4C5C 769 GFLOAT_TO_DOUBL: \$BASS\$MAT_TRN G, D
4E44 770

BASS\$MAT_TRN
1-013

B 3

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 57
(5)

4E44 772 GFLOAT_TO_GFLOA: \$BASS\$MAT_TRN G, G
502D 773

BASS\$MAT_TRN
1-013

C 3

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATRN.MAR;1 Page 58
(5)

502D 775 GFLOAT_TO_HFLOA: \$BASS\$MAT_TRN G, H
521A 776

521A 778 ;+
 521A 779 ; Source array is a hfloat array. Now differentiate on the destination type.
 521A 780 ;-
 521A 781
 05 06 55 5B D0 521A 782 HFLOAT: MOVL R11, R5 ; recover original pointer
 8F 521D 783 5\$: CASEB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_B, #<DSC\$K_DTYPE_D - DSC\$K_DTYPE_B>
 002D 5222 784 1\$: .WORD HFLOAT_TO_BYTE-1\$; code for byte dtype
 0215 5224 785 .WORD HFLOAT_TO_WORD-1\$; code for word dtype
 03FD 5226 786 .WORD HFLOAT_TO_LONG-1\$; code for long dtype
 AEA1 5228 787 .WORD ERR_DATTYPERR-1\$; quad not supported
 05E5 522A 788 .WORD HFLOAT_TO_FLOAT-1\$; code for float dtype
 07CD 522C 789 .WORD HFLOAT_TO_DOUBL-1\$; code for double dtype
 1B 02 A5 91 522E 790
 03 12 5232 791 CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_G
 09B6 31 5234 792 BNEQ 2\$
 5237 793 BRW HFLOAT_TO_GFLOA
 1C 02 A5 91 5237 795 2\$: CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_H
 03 12 523B 796 BNEQ 3\$
 0B9A 31 523D 797 BRW HFLOAT_TO_HFLOA
 18 02 A5 91 5240 798
 06 12 5244 800 CMPB DSC\$B_DTYPE(R5), #DSC\$K_DTYPE_DSC
 55 04 A5 D0 5246 801 BNEQ 4\$
 D1 11 524A 802 MOVL 4(R5), R5 ; R5 <-- addr of descriptor
 524C 803 BRB 5\$; CASE again on dtype in desc
 AE74 31 524C 804 4\$: BRW ERR_DATTYPERR ; unsupported dtype
 524F 805
 524F 806 ;+
 524F 807 ; Now type of source and destination arrays are known. Use the macro to
 524F 808 ; generate the code for each case
 524F 809 ;-
 524F 810

BASS\$MAT_TRN
1-013

E 3

BASS\$MAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATRN.MAR;1 Page 60
(5)

524F 812 HFLOAT_TO_BYTE: \$BASS\$MAT_TRN H, B
5437 813

BASSMAT_TRN
1-013

F 3

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATTRN.MAR;1 Page 61 (5)

5437 815 MFLOAT_TO_WORD: \$BASSMAT_TRN H, W
561F 816

BASSMAT_TRN
1-013

G 3

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMATTRN.MAR;1 Page 62
(5)

561F 818 HFLOAT_TO_LONG: \$BASSMAT_TRN H, L
5807 819

BASSMAT_TRN
1-013

H 3
BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMATTRN.MAR;1 Page 63 (5)
5807 821 HFLOAT_TO_FLOAT: \$BASSMAT_TRN H, F
59EF 822

BASSMAT_TRN
1-013

I 3

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATRN.MAR;1 Page 64 (5)

59EF 824 HFLOAT_TO_DOUBL: \$BASSMAT_TRN H, D
5BED 825

BASSMAT_TRN
1-013

J 3

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATRN.MAR;1 Page 65 (5)

5BED 827 HFLOAT_TO_GFLOA: \$BASSMAT_TRN H, G
5DDA 828

BASSMAT_TRN
1-013

K 3

BASSMAT_TRN - Transpose one matrix into 15-SEP-1984 23:55:09 VAX/VMS Macro V04-00
6-SEP-1984 10:31:25 [BASRTL.SRC]BASMMATRN.MAR;1 Page 66 (5)

5DDA 830 HFLOAT_TO_HFLOA: \$BASSMAT_TRN H, H
5FC3 831
5FC3 832 .END

; end of BASSMAT_TRN

BASS\$SCALE_R1	*****	X	00	DSC\$L_L1_1	= 00000018
BASS\$STOP	*****	X	00	DSC\$L_L1_2	= 0000001C
BASS\$FETCH_BFA	*****	X	00	DSC\$L_L2_2	= 00000024
BASS\$FET_F_A_B_R8	*****	X	00	DSC\$L_LOGUNIT	= FFFFFFFC
BASS\$FET_F_A_D_R8	*****	X	00	DSC\$L_M1	= 00000014
BASS\$FET_F_A_F_R8	*****	X	00	DSC\$L_M2	= 00000018
BASS\$FET_F_A_G_R8	*****	X	00	DSC\$L_U1_1	= 0000001C
BASS\$FET_F_A_H_R8	*****	X	00	DSC\$L_U1_2	= 00000020
BASS\$FET_F_A_L_R8	*****	X	00	DSC\$L_U2_2	= 00000028
BASS\$FET_F_A_W_R8	*****	X	00	DSC\$V_FL_BOUNDS	= 00000007
BASS\$ARGDONMAT	*****	X	00	DSC\$W_LENGTH	= 00000000
BASS\$DATATYPEERR	*****	X	00	DTYPE	= 0000000E
BASS\$ILLOPE	*****	X	00	ERR_ARGDONMAT	= 00000024 R 02
BASS\$MATDIMERR	*****	X	00	ERR_DATATYPEERR	= 000000C3 R 02
BASS\$MATREDIM	*****	X	00	ERR_ILLOPE	= 00000031 R 02
BASS\$MATTRN	00000000	RG	02	ERR_MATDIMERR	= 00000017 R 02
BASS\$STORE_BFA	*****	X	00	FLOAT	= 00002941 R 02
BASS\$STO_F_A_B_R8	*****	X	00	FLOAT_TO_BYTE	= 00002976 R 02
BASS\$STO_F_A_D_R8	*****	X	00	FLOAT_TO_DOUBL	= 000030F7 R 02
BASS\$STO_F_A_F_R8	*****	X	00	FLOAT_TO_FLOAT	= 00002F19 R 02
BASS\$STO_F_A_G_R8	*****	X	00	FLOAT_TO_GFLOA	= 000032EE R 02
BASS\$STO_F_A_H_R8	*****	X	00	FLOAT_TO_HFLOA	= 000034D5 R 02
BASS\$STO_F_A_L_R8	*****	X	00	FLOAT_TO_LONG	= 00002D38 R 02
BASS\$STO_F_A_W_R8	*****	X	00	FLOAT_TO_WORD	= 00002B57 R 02
BYTE	00000000	R	02	GFLOAT	= 00004487 R 02
BYTE_TO_BYTE	00000105	R	02	GFLOAT_TO_BYTE	= 000044BC R 02
BYTE_TO_DOUBLE	00000886	R	02	GFLOAT_TO_DOUBL	= 00004C5C R 02
BYTE_TO_FLOAT	000006A5	R	02	GFLOAT_TO_FLOAT	= 00004A74 R 02
BYTE_TO_GFLOAT	00000A7D	R	02	GFLOAT_TO_GFLOA	= 00004E44 R 02
BYTE_TO_HFLOAT	00000C64	R	02	GFLOAT_TO_HFLOA	= 0000502D R 02
BYTE_TO_LONG	000004C4	R	02	GFLOAT_TO_LONG	= 0000488C R 02
BYTE_TO_WORD	000002E3	R	02	GFLOAT_TO_WORD	= 000046A4 R 02
CLASS	= 0000000F			HFLOAT	= 0000521A R 02
DATA	= 00000014			HFLOAT_TO_BYTE	= 0000524F R 02
DEST_MATRIX	= 00000008			HFLOAT_TO_DOUBL	= 000059EF R 02
DOUBLE	000036BC	R	02	HFLOAT_TO_FLOAT	= 00005807 R 02
DOUBLE_TO_BYTE	000036F1	R	02	HFLOAT_TO_GFLOA	= 00005BED R 02
DOUBLE_TO_DOUBL	00003EB9	R	02	HFLOAT_TO_HFLOA	= 00005DDA R 02
DOUBLE_TO_FLOAT	00003CC7	R	02	HFLOAT_TO_LONG	= 0000561F R 02
DOUBLE_TO_GFLOAT	00004097	R	02	HFLOAT_TO_WORD	= 00005437 R 02
DOUBLE_TO_HFLOAT	0000428F	R	02	INIT_S0BS	= 0000003E R 02
DOUBLE_TO_LONG	00003AD5	R	02	INIT_SUBS_2	= 0000006B R 02
DOUBLE_TO_WORD	000038E3	R	02	LONG	= 00001BC6 R 02
DSC\$A_X0	= 00000010			LONG_TO_BYTE	= 00001BFB R 02
DSC\$A_POINTER	= 00000004			LONG_TO_DOUBLE	= 0000237C R 02
DSC\$B_AFLAGS	= 0000000A			LONG_TO_FLOAT	= 0000219B R 02
DSC\$B_CLASS	= 00000003			LONG_TO_GFLOAT	= 00002573 R 02
DSC\$B_DIMCT	= 0000000B			LONG_TO_HFLOAT	= 0000275A R 02
DSC\$B_DTYPE	= 00000002			LONG_TO_LONG	= 00001FBD R 02
DSC\$K_CLASS_A	= 00000004			LONG_TO_WORD	= 00001DDC R 02
DSC\$K_CLASS_BFA	= 000000BF			LOOP_1ST_SUBBB	= 00000105 R 02
DSC\$K_DTYPE_B	= 00000006			LOOP_1ST_SUBBD	= 00000886 R 02
DSC\$K_DTYPE_D	= 0000000B			LOOP_1ST_SUBBF	= 000006A5 R 02
DSC\$K_DTYPE_DSC	= 00000018			LOOP_1ST_SUBBG	= 00000A7D R 02
DSC\$K_DTYPE_G	= 0000001B			LOOP_1ST_SUBBH	= 00000C64 R 02
DSC\$K_DTYPE_H	= 0000001C			LOOP_1ST_SUBBL	= 000004C4 R 02
DSC\$L_BYTEOF	= FFFFFFF8			LOOP_1ST_SUBBW	= 000002E3 R 02

BASSMAT TRN
Symbol Table

LOOP_1ST_SUBDB	000036F1	R	02	LOOP_2ND_SUBFD	000030FA	R	02
LOOP_1ST_SUBDD	00003EB9	R	02	LOOP_2ND_SUBFF	00002F1C	R	02
LOOP_1ST_SUBDF	00003CC7	R	02	LOOP_2ND_SUBFG	000032F1	R	02
LOOP_1ST_SUBDG	00004097	R	02	LOOP_2ND_SUBFH	000034D8	R	02
LOOP_1ST_SUBDH	0000428F	R	02	LOOP_2ND_SUBFL	00002D3B	R	02
LOOP_1ST_SUBDL	00003AD5	R	02	LOOP_2ND_SUBFW	00002B5A	R	02
LOOP_1ST_SUBDW	000038E3	R	02	LOOP_2ND_SUBGB	000044BF	R	02
LOOP_1ST_SUBFB	00002976	R	02	LOOP_2ND_SUBGD	00004C5F	R	02
LOOP_1ST_SUBFD	000030F7	R	02	LOOP_2ND_SUBGF	00004A77	R	02
LOOP_1ST_SUBFF	00002F19	R	02	LOOP_2ND_SUBGG	00004E47	R	02
LOOP_1ST_SUBFG	000032EE	R	02	LOOP_2ND_SUBGH	00005030	R	02
LOOP_1ST_SUBFH	000034D5	R	02	LOOP_2ND_SUBGL	0000488F	R	02
LOOP_1ST_SUBFL	00002D38	R	02	LOOP_2ND_SUBGW	000046A7	R	02
LOOP_1ST_SUBFW	00002B57	R	02	LOOP_2ND_SUBHB	00005252	R	02
LOOP_1ST_SUBGB	000044BC	R	02	LOOP_2ND_SUBHD	000059F2	R	02
LOOP_1ST_SUBGD	00004C5C	R	02	LOOP_2ND_SUBHF	0000580A	R	02
LOOP_1ST_SUBGF	00004A74	R	02	LOOP_2ND_SUBHG	00005BF0	R	02
LOOP_1ST_SUBGG	00004E44	R	02	LOOP_2ND_SUBHH	00005DDD	R	02
LOOP_1ST_SUBGH	0000502D	R	02	LOOP_2ND_SUBHL	00005622	R	02
LOOP_1ST_SUBGL	0000488C	R	02	LOOP_2ND_SUBHW	0000543A	R	02
LOOP_1ST_SUBGW	000046A4	R	02	LOOP_2ND_SUBLB	00001BFE	R	02
LOOP_1ST_SUBHB	0000524F	R	02	LOOP_2ND_SUBLD	0000237F	R	02
LOOP_1ST_SUBHD	000059EF	R	02	LOOP_2ND_SUBLF	0000219E	R	02
LOOP_1ST_SUBHF	00005807	R	02	LOOP_2ND_SUBLG	00002576	R	02
LOOP_1ST_SUBHG	00005BED	R	02	LOOP_2ND_SUBLH	0000275D	R	02
LOOP_1ST_SUBHH	00005DDA	R	02	LOOP_2ND_SUBLL	00001FC0	R	02
LOOP_1ST_SUBHL	0000561F	R	02	LOOP_2ND_SUBLW	00001DDF	R	02
LOOP_1ST_SUBHW	00005437	R	02	LOOP_2ND_SUBWB	00000E83	R	02
LOOP_1ST_SUBLB	00001BFB	R	02	LOOP_2ND_SUBWD	00001604	R	02
LOOP_1ST_SUBLD	0000237C	R	02	LOOP_2ND_SUBWF	00001423	R	02
LOOP_1ST_SUBLF	0000219B	R	02	LOOP_2ND_SUBWG	000017FB	R	02
LOOP_1ST_SUBLG	00002573	R	02	LOOP_2ND_SUBWH	000019E2	R	02
LOOP_1ST_SUBLH	0000275A	R	02	LOOP_2ND_SUBWL	00001242	R	02
LOOP_1ST_SUBLL	00001FBD	R	02	LOOP_2ND_SUBWW	00001064	R	02
LOOP_1ST_SUBLW	00001DDC	R	02	LOWER_BND1	= 00000004		
LOOP_1ST_SUBWB	00000E80	R	02	LOWER_BND2	= 00000000		
LOOP_1ST_SUBWD	00001601	R	02	MTH\$DINT_R4	***** X 00		
LOOP_1ST_SUBWF	00001420	R	02	POINTER	= 00000010		
LOOP_1ST_SUBWG	000017F8	R	02	SEPARATE_DTYPES	= 00000091 R 02		
LOOP_1ST_SUBWH	000019DF	R	02	SF\$L_SAVE_FP	= 0000000C		
LOOP_1ST_SUBWL	0000123F	R	02	SRC_MATRIX	= 00000004		
LOOP_1ST_SUBWW	00001061	R	02	STR_LEN	= 0000000C		
LOOP_2ND_SUBBB	00000108	R	02	UPPER_BND1	= 00000008		
LOOP_2ND_SUBBD	00000889	R	02	VALUE_DESC	= 0000000C		
LOOP_2ND_SUBBF	000006A8	R	02	VIRTUAL	= 00000057 R 02		
LOOP_2ND_SUBBG	00000A80	R	02	WORD	= 00000E4B R 02		
LOOP_2ND_SUBBH	00000C67	R	02	WORD_TO_BYTE	= 00000E80 R 02		
LOOP_2ND_SUBBL	000004C7	R	02	WORD_TO_DOUBLE	= 00001601 R 02		
LOOP_2ND_SUBBW	000002E6	R	02	WORD_TO_FLOAT	= 00001420 R 02		
LOOP_2ND_SUBDB	000036F4	R	02	WORD_TO_GFLOAT	= 000017F8 R 02		
LOOP_2ND_SUBDD	00003EBC	R	02	WORD_TO_HFLOAT	= 000019DF R 02		
LOOP_2ND_SUBDF	00003CCA	R	02	WORD_TO_LONG	= 0000123F R 02		
LOOP_2ND_SUBDG	0000409A	R	02	WORD_TO_WORD	= 00001061 R 02		
LOOP_2ND_SUBDH	00004292	R	02				
LOOP_2ND_SUBDL	00003AD8	R	02				
LOOP_2ND_SUBDW	000038E6	R	02				
LOOP_2ND_SUBFB	00002979	R	02				

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_BASS\$CODE	00005FC3 (24515.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	28	00:00:00.08	00:00:00.31
Command processing	102	00:00:00.66	00:00:02.89
Pass 1	1020	00:00:40.75	00:01:26.26
Symbol table sort	6	00:00:01.85	00:00:09.19
Pass 2	635	00:00:09.69	00:00:32.14
Symbol table output	89	00:00:00.22	00:00:02.31
Psect synopsis output	7	00:00:00.04	00:00:00.21
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1890	00:00:53.31	00:02:13.34

The working set limit was 750 pages.

297708 bytes (582 pages) of virtual memory were used to buffer the intermediate code

There were 60 pages of symbol table space allocated to hold 369 non-local and 824 local symbols.

832 source lines were read in Pass 1, producing 75 object records in Pass 2.

32 pages of virtual memory were used to define 11 macros.

Macro library statistics

Macro library name

Macros defined

\$255\$DUA28:[BASRTL.OBJ]BASRTL.MLB;1
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2
TOTALS (all libraries)

2

5

7

493 GETS were required to define 7 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE={GLOBAL - TRACEBACK}/LIS=LIS\$:BASMAITRN/OBJ=OBJ\$:BASMAITRN MSRC\$:BASMAITRN/UPDATE={ENH\$:BASMAITRN}+1

0027 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

BASMATSUB
LIS

BASMATSCA
LIS

BASMATTRN
LIS

0028 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

BASMD
LIS

BASMD01
LIS

BASNOTIMP
LIS

BASMOVEAR
LIS

BASMSGDEF
LIS

BASMSGGEN
LIS

BASONECHR
LIS

BASMOVE
LIS

BASNUM
LIS

BASNAMEAS
LIS

BASNUM1
LIS